EUROPEAN PATENT SPECIFICATION

(54) Digital data message transmission networks and the establishing of communication paths therein.

(73) Proprietor: International Business Machines
Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Brice, Frank William, Jr.
1C Stony Run East
Kingston New York 12401(US)
Inventor: Weingarten, Robert Allen
1 Adam Court
Highland Mills New York 10930(US)

(74) Representative: Moss, Robert Douglas
IBM United Kingdom Limited Intellectual
Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

IBM SYSTEMS JOURNAL, vol. 22, no. 4, 1983, pages 417-434, Armonk, New York, US; J.M. JAFFE et al.: "SNA routing: Past, present, and possible future"

IBM TECHNICAL DISCLOSURE BULLETIN, vol. 22, no. 11, April 1980, pages 5209-5212, New York, US; K. MARUYAMA: "Session failure notification using non-bidirectional explicit paths"

PROCEEDINGS OF THE SEVENTH INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATION, Sydney, 30th October - 2nd November 1984, pages 730-735, Elsevier Science Publishers B.V. (North-Holland), Amsterdam, NL; J.L. EISENBIES et al.: "An automatic topology update scheme for SNA networks"

Description

The present invention relates to digital data message networks and the establishing of communication paths therein in which the communication path, or route, that a message takes through the network is computed dynamically, that is, the path is determined anew each time a communication session between two users of the network is attempted to be established.

Communication networks are arrangements of computers, communication controllers and associated peripheral equipment and links which allow "end users" located at remote locations in the network to establish and maintain communication of information. An end user in this context may be a human user at some type of terminal, or an application program running on a host computer or on some type of intelligent device controller (ie, workstation, personal computer, display controller, etc). Figure 1 illustrates a small communication network 10. In it, two IBM System 370 host computers 12, 14 and three IBM 3725 Communications Controllers 16, 18, 20 are interconnected, supporting five terminals 22, 24, 26, 28, 30 connected as shown. Communication links 32, 34, 46, 38, 40, 44, interconnect the computers 12, 14 and controllers 16, 18, 20 as shown.

Communication between, say, terminal 30 and host computer 12 in Figure 1 would be possible over several different paths. In order to visualise the communication in a network, a convention has been adopted in which the computers and communication controllers in a network are referred to collectively as nodes and the connections between nodes as links. End-user devices, whether displays, printers, etc are referred to collectively as terminals. Figure 2 depicts a node and link diagram of the network 10 shown in Figure 1, with like elements in Figure 2 having the same reference numerals as in Figure 1, but primed. Terminals are not shown in the diagram, since they are not directly involved in the communication establishment and control process to which the present invention is related.

A conversation, called a session, between an end user at terminal 30 and an application program in host computer 12 in Figure 1 would involve communication between nodes 12 and 18, or 12' and 18' in Figure 2. Several different paths are possible for such a session. For example, communication could be simply via link 34', or the path could involve links 32' and 40' and node 16'. Alternatively, the path could involve links 32', 36' and 38' and nodes 16' and 14'. The choices increase with the size and complexity of the network.

Except in very limited circumstances in which the network is fully connected, the information input at an origin node (supporting one end user) travels through one or more intermediate nodes before reaching the destination node (supporting the other end user). Information transmitted by one node and directed to another node must travel over the links interconnecting the nodes. In some networks, all information transmitted between two end users during a specific communication session traverses the same path, called in that case a route. The mechanism used to create the route, hereafter called the route mechanism, is dependent on a specific network architecture.

Most network routing mechanisms use routing tables in each node (origin, intermediate, and destination) to forward messages to the next node. Routing mechanisms may vary but several use an index into a routing table (the index being either a route identifier or a combination of source and/or destination node identifiers) to specify to which outbound link, and therefore to which next adjacent network node, the message is to be sent. Naturally when the message reaches the destination node, but is instead processed by the end user.

The routing tables in each network node can be established either statically or dynamically. In the static case, the routing definitions are fixed when network operating is started. The static mechanism is not relevant to the invention and will therefore not be discussed.

In dynamic routing, a route creation technique is employed that dynamically creates an end-to-end route, from the source node to a destination node, which can be used during a specific communication session. This end-to-end route remains defined and active until all sessions using the route terminate. Then the route can be eliminated to free network resources, such as routing table entries to be used for other dynamically created routes.

Several methods can be used to create dynamic routes within networks. One such method, disclosed in Proceedings of the Seventh International Conference on Computer Communication, Sydney, 30.10.-2.11.1984 Elsevier Pub., Amsterdam, NL; J.H. Jaffe et al.; "Applying dynamic routing for a general network architecture", pages 716-721, employs a topology data base which contains the identity of all physical nodes and links in the network, including their connectivity, status, and physical characteristics to build a route dynamically. This method selects appropriate nodes and links based on their recorded status and creates a message, hereafter called a ROUTE SETUP message, containing a list of the nodes and links to be used in the route. The ROUTE SETUP message traverses each node in the list, allowing each node to

3

build an entry in its routing table so that subsequent messages for a session assigned to the route can traverse the same physical nodes identified in the ROUTE-SETUP message. Both forward and backward pointers are placed in the node routing table.

Routing in communication networks is discussed by Jaffe et al in "SNA Routing: Past, Present and Possible Future" appearing in the IBM Systems Journal, Vol. 22, No 4 (1983), Tymes in "Routing and Flow Control in TYMNET" appearing in the IEEE Transactions on Communications, Vol. COM-29, No 4, (1981), and by McQuillan et al in "The New Routing Algorithm for ARPANET" appearing in the IEEE Transactions on Communications, Vol. COM-28, No 5, (1980).

Clearly, it is important for the topology data base to have current information as to which nodes and lines are available for communication. Otherwise, attempts might be made to create routes which have unavailable nodes or links, and several route creation attempts might be necessary before an available route is actually found. This represents an unnecessary waste of network resources. Other than by manual, operator input, the reporting of the initial status, failures and reactivations of the network nodes and links, hereafter called network elements, has been done by the broadcast of status changes by adjacent nodes. The Tymes and McQuillan et al articles mentioned above describe such broadcast update methods. These broadcast status updates have the advantage of not requiring operator intervention. However, they have the problem, especially in large and complex networks, of considerably increasing network overhead traffic. In some situations, it is preferable to accept less than perfect knowledge of the network status in order to reduce this overhead. In fact, it is necessary in such networks to have some mechanism to "kill" the broadcast message which propagates throughout the system, or else the message reverberates throughout the network forever. Typical mechanisms include the incorporation of a cutoff time, after which the message is discarded from the network. However, even with such limiting mechanisms, status update messages occupy a large proportion of the overall network message-handling capability, such that there is less capacity available for users messages.

Thus, it is apparent that there is a need for a means for updating a topology data base in a dynamic routing environment such that there are significantly reduced demands on the message handling capability of the network as compared with prior art schemes for updating topology data bases.

Accordingly, the present invention provides a digital data message transmission network comprising nodes connected by links, wherein a message travels between a source node and a destination node along a route comprising a sequence of nodes and connecting links, said network including means for maintaining a topology data base recording the potentially available communication routes through the network and the status of the network nodes and links; each node being adapted to maintain a routing table and: when acting as a source node, obtain a route from the topology database and send a ROUTE SET-UP message along the route to thereby establish the route; respond to a ROUTE SET-UP message by recording an entry in its routing table with the identity of the route and the links and/or nodes therein that are immediately adjacent to that particular node; said network being characterised in that each node is further adapted to: detect any failure of an immediately adjacent node or link in the route either whilst attempting to establish the route or after the route has been established, and in response thereto (i) record the status of the route as inoperative in the respective entry in the node's routing table, said entry remaining substantially intact; and (ii) transmit a ROUTE FAILURE message identifying the failed node or link along the route in the opposite direction to the failed node or link; respond to a ROUTE FAILURE message by (i) forwarding said ROUTE FAILURE message along the route in said opposite direction; and by
(ii) further recording the status of the route as inoperative in the respective entry in the node's routing table, said entry remaining substantially intact; and when acting as a source or destination node, additionally forward said ROUTE FAILURE message to the topology data base for updating the recorded status of the network nodes and links.

The present invention further provides a method of controlling a digital data message transmission network as set out in claim 5.

Such an arrangement greatly reduces the overall status message traffic in even a complex communication network by ensuring that only the specific nodes involved in the use of an element are informed of the current status of that element.

The present invention will be described further by way of example with reference to preferred embodiments of the invention, as illustrated in the accompanying drawings, in which:

FIG. 1 is a block diagram of a simple mesh type communication network;
FIG. 2 is a schematic representation of the network shown in FIG. 1;
FIG. 3 is a schematic representation of a topology data base route table used by the network of Fig. 1;
FIG. 4 is a schematic representation of a route statement;
FIG. 5 is a schematic representation of a ROUTE REQUEST message;

FIG. 7 is a schematic representation of a ROUTE-SETUP message;

FIG. 8 is a schematic representation of a routing table;

FIG. 9 is a schematic representation of a ROUTE FAILURE message; and

FIG. 10 is a schematic representation of a network with a topology data base.

5 In the drawings, like elements are designated with similar reference numbers, and identical elements in different specific embodiments are designated by identical reference numbers.

The present invention incorporates a method and apparatus which can be used to dynamically report the status of the network nodes and links to topology data bases so that ROUTE-SETUP messages can be created efficiently with knowledge of the physical route element status. The description that follows is 10 applicable to an IBM SNA network comprising, for example, IBM System 370 computers and IBM 3725 Communication Controllers as nodes, and suitable communication links for network interconnection. However, the principles and protocols are presented with sufficient generality to enable application to any communication network having dynamic routing that relies on one or more topology data bases for route creation, and local routing tables at each node for the establishment and maintenance of communication 15 sessions.

In the preferred embodiment, each topology data base is initially set up with information as to all of the nodes and links, and their potential connectivity to adjacent nodes. This information is entered by way of conventional input statements either interactively or via batch definitions. The link and node statements further specify certain characteristics, such as line(link) transmission speed, node buffer size, and element 20 data transmission security, which are used by end users to request a route. These characteristics are set forth below under RULES OF OPERATION, DEFINITIONS & STATES, part a) Data Types. Additionally, each network element in the topology data base is initially marked as "assumed operative", or available. This initial status has nothing to do with the actual status of the corresponding elements, but allows the route selection technique to select the element for a route setup attempt and dynamically ascertain its 25 actual operative or inoperative status via the feedback mechanism. The information is stored conventionally, in such a way as to be available for scanning and comparing when a route request is received.

This information is entered for 1) routes, 2) links, and 3) nodes. Thus, assuming the interactive input method (as opposed to batch), a route statement is used to enter information defining a route between two nodes between which communication is to be allowed. This statement specifies the links and nodes, in 30 order, in that route, and an identifier (ID) for that route. For each pair of nodes that may communicate, as many route statements as there are distinct routes between those nodes may be entered. Figure 3 illustrates the topology data base entries for the potential routes between nodes 16' and 14' in Figure 2. In this case there are three such routes: 50, 52, and 54. Figure 4 illustrates the form of a route statement 58. Link and node names 60 are placed in the sequence in which a message traverses that route, starting with 35 the origin node and ending with the destination node. The statement type 62 identifies the statement as a route statement.

Figure 5 illustrates a link statement 64 and a node statement 66, showing the various fields 68, 70 identifying the element and containing data for the aforementioned characteristics. The statement type 72, 74, identifies the statements as link or node statements, respectively. The information entered via the route, 40 link and node statements is conventionally stored in such a way as to be available for scanning and comparing when a route request is received, so as to, first, retrieve all routes identified between the specified end nodes, and, second, find the best match of characteristics for the elements in the selected routes against the specified characteristics and thereby select the best route.

When an end user desires to communicate with another end user, a session is requested of the control 45 program resident in the origin node. The session request includes the names of the source and destination nodes supporting the end users and a class of service. The class of service is a conventional shorthand for a set of the aforementioned characteristics of the route to be selected. In the preferred embodiment, the information in the session request is forwarded in a message, hereafter called a ROUTE REQUEST message, to the nearest available topology data base for the selection of a route. Figure 6 illustrates a 50 ROUTE REQUEST message 76. It comprises a request code 78, identifying the message as a ROUTE REQUEST, the origin node name 79, the destination node name 80, and the class of service name 82.

Upon receipt of the ROUTE REQUEST message, the topology data base selects a potential route whose elements conform to the characteristics identified in the specified class of service. If a route with the same exact set of elements is already in use with the same characteristics, the session will be assigned to 55 the already created route. If a route has not already been created, the topology data base is scanned to select the best route between the source and destination nodes, based on the match of actual element characteristics against the characteristics requested. Only nodes and links which are assumed or known to be operative can be selected in this process. The selected route is then described in a message containing

a list of nodes and links comprising the route, along with a unique route identifier (route ID) for that route, hereafter called a ROUTE-SETUP message. Figure 7 illustrates a ROUTE-SETUP message 84. The request code 86 identifies the message as a ROUTE-SETUP message. The route ID 87 identifies the route. The node names and link names 88 appear in the sequence a message travels on the route, as in the route
5 statement 58 (Figure 4, supra).

Route elements are assumed initially to be operative, because when the topology data base is activated, the status of each network element is unknown. Rather than initially obtaining the status of each element defined in the topology data base through operator input, exchanges with other data bases, or queries of each network element, the status is learned as sessions are requested by using the preferred
10 embodiment of the present invention herein described.

As the ROUTE-SETUP message traverses the network, each active node creates routing table entries in conventional fashion for both directions of the communication. That is, when the ROUTE-SETUP message is processed, an entry is made in the routing table, containing the route ID (including origin and destination node names or addresses) and the outbound link address. As is known, this permits quick routing of
15 messages thereafter via the route ID accompanying the message. When a message arrives with this information, a simple table scan yields the outbound link address, which is sufficient for sending the message on its way.

In an alternative embodiment, the route ID is reduced to a single Local Path Identifier (LPID) in each node, which may be a sequentially selected number starting with zero that is assigned to a route as the
20 ROUTE-SETUP message for that route arrives. The LPID is entered into the table, and sent back to the previous node for entry into that node's routing table. Each routing table, in this embodiment, has the LPID it assigns, the outbound link to the next node, and the LPID assigned by the next node in the route. When a message arrives at a node in this embodiment, the LPID for that node is extracted from the header and used to locate the table entry for that session. The next node's LPID, located in the table, is then exchanged
25 for the incoming LPID. The message is then sent on to the next node via the assigned outbound link, and so forth. Figure 8 illustrates a routing table 90 including a unidirectional entry 88 for a hypothetical route, using the LPID method. As shown, there are three elements to an entry, an LPID 94, the outbound link ID 96, and the LPID for the next node 98.

After the information necessary for the routing table entry is extracted from the ROUTE-SETUP
30 message, it is sent on to the next element, and so on, to create all of the necessary table entries for the route. If all nodes and links in the end-to-end route are active, the ROUTE-SETUP will be successful and an end-to-end route will be created.

Each routing table entry in each node traversed is marked as operative. If a node cannot forward the ROUTE-SETUP message to the next node because either the link to that node or the node itself is
35 inoperative, then a ROUTE FAILURE message identifying the inoperative link or node is generated and returned to the route origin for forwarding to the topology data base. The inoperative status is marked in the topology data base and that network element is not used in subsequent ROUTE-SETUP messages until its status changes.

As the ROUTE FAILURE message returns through each node identified in the ROUTE-SETUP, each
40 node's routing table entry remains intact, with the status of the route being marked as inoperative. These table entries are used for the feedback mechanism to the end node (and its topology data base) when the failed element is re-activated. The ROUTE FAILURE message is transmitted by following the node routing table entries set up from the failed ROUTE-SETUP message. Figure 9 illustrates a ROUTE FAILURE message 100 for an LPID-based network. It includes a Request Code 102 identifying the message as a
45 ROUTE FAILURE message, the node ID of the node detecting the failure 104, Failed Element ID 106, and Next LPID 108.

When the node that reported the ROUTE FAILURE (the "adjacent node" detects that the inoperative element subsequently has become operative, a ROUTE ELEMENT OPERATIVE message is created and sent to the origin node for forwarding to its topology data base. The topology data base then resets that
50 element's status entry to operative. The element is then available for subsequent ROUTE-SETUP attempts. The ROUTE ELEMENT OPERATIVE message is identical to a ROUTE FAILURE message (100, Fig. 9), except that the Request Code identifies it as a ROUTE ELEMENT OPERATIVE message.

If more than one route had attempted to use the inoperative element, more than one set of routing table entries would have been created in the adjacent node and for as many of those sets of entries as had been
55 created. ROUTE ELEMENT OPERATIVE messages would be created and sent back to the corresponding origin nodes for resetting of their topology data base entries for that element to operative.

This notification flows to the route origin node by following the node routing table entries set up from the failed ROUTE-SETUP message, in the same manner as the ROUTE FAILURE message.

The same process is used after an end-to-end route is successfully established and a failure occurs on a link or node in the route. The same ROUTE FAILURE message is returned, in this case to both end nodes, the origin and the destination node, for forwarding to the topology data base of each end node. The ROUTE FAILURE message identifies the failing element, and as in the initial case of failure to set up the
5 route, the routing table entries in the route segments to the end nodes remain intact as the message travels back to the end node. As in the previous case, when the failed element becomes operative, the node routing table entries are used by ROUTE ELEMENT OPERATIVE messages (one in each direction around the point of failure) to follow the route back to the end node for forwarding to the topology data base(s) to indicate the operative status of the network element.
10 In either case, as the ROUTE ELEMENT OPERATIVE messages flow to the topology data base(s), the routing table entries are removed from the nodes along the route to recover the routing table space just as if the end-to-end route were being normally deleted after the termination of all sessions using the route.

The feedback mechanism functions in the face of both single point and multiple point failures along the route, as described in greater detail below.
15 The basic elements of the feedback mechanism of the preferred embodiment can be described by a set of rules, definitions, states and protocols which may be applied to any communication network having dynamic routing of the type herein described. The protocols are described in four operational cases.

RULES OF OPERATION, DEFINITIONS & STATES
20

1. Topology Data Base. There will exist one or more topology data bases which contain network node, connectivity and status data which can be used to select and end-to-end route through a communication network.

For each route element (that is, for each network node, which can be either source, destination, or
25 intermediate node host or communication processor; and for each link, which can be either teleprocessing link, channel connection, local teleprocessing loop, Local Area Network, fibre optic line, or any other medium which can connect two host or communication processor nodes), there exists a set of parameters as follows:

30

Element	Identifiers	Connectivity	Characteristics	Status
Node	Node Name	Adjacent	Message Storage	Operative
	Node Address	nodes	Capacity	Inoperative
			Processing Power	Unknown/
			Security	(Assumed
				Operative)
Link	Link Name	Adjacent	Bandwidth	Operative
	Link Address	nodes	Reliability	Inoperative
			Security	Unknown/
				(Assumed
				Operative)

50 These characteristics are well known in the art.

Various methods can be utilised to create the entries and element knowledge in the topology data base. They include operator input, a system generation process, automatic element identification at element activation, etc.

2. Route Selection Algorithm. The algorithm(s) utilised to select a route (ie, secure, least delay, greatest
55 bandwidth, etc) should not affect the operability of the invention in a network, provided the selection algorithm adheres to the rules for creation of the selected route list and element status described above.

3. Route Request Mechanism. A mechanism is provided to allow a user to request an end-to-end route from source to destination nodes through the network. This request is sent to a topology data base with

7

an indication as to the characteristics (secure, least delay, etc) desired. The request initiates the route selection process in the topology data base and creates a ROUTE-SETUP message.

4. ROUTE-SETUP Message. This is a message which contains the list of nodes and links selected to create the end-to-end route from source to destination nodes. The ROUTE-SETUP message traverses the network, from the source to the destination, node to link to node to link, etc, in the order specified in the ROUTE-SETUP message. As it traverses the proposed route, routing tables are created in each node traversed, the forward and backward adjacent node connections being indicated by entries in the routing table. The ROUTE-SETUP can have one of two replies: ROUTE-SETUP-REPLY message or ROUTE-FAILURE message.

5. ROUTE-SETUP-REPLY Message This message is issued from the destination back to the source on a successful creation of a route.

6. The ROUTE FAILURE Message is returned for a failed ROUTE-SETUP attempt, and for the failure of an element in a successfully-created route. In the case of a failed ROUTE-SETUP attempt, when a network node cannot continue to forward the ROUTE-SETUP message to its neighbour, it creates a ROUTE-FAILURE message. This message, illustrated in Figure 9, is returned to the source to be forwarded to the topology data base. This ROUTE-FAILURE message indicates which node detected the inoperative network resource. The topology data base records the status of the inoperative element (ie, L3) and does not reuse this element for other ROUTE-SETUP attempts until subsequently informed that the element is operative by the ROUTE-ELEMENT-OPERATIVE message. In the case of the failure of an element on an established route, the same ROUTE-FAILURE message is created, and for the same purpose, but in this case, it is sent to both participating end nodes.

7. The ROUTE-ELEMENT-OPERATIVE Message is created by a node when it detects the reactivation of an inoperative element. The ROUTE-ELEMENT-OPERATIVE message is created and sent provided that a ROUTE-SETUP message has flowed prior to the element changing from inoperative to operative. The ROUTE-ELEMENT-OPERATIVE message traverses the network back to the origin node by using the routing table entries from the earlier ROUTE-SETUP MESSAGE.

8. ROUTE-DELETE Message. This message is sent on the route to signal that the route is no longer needed. It is created by the route end-points (either source or destination node) when all sessions assigned to a specific end-to-end route have ended. As the ROUTE-DELETE message traverses the network, each node in its path frees the routing table entries assigned to that specific route and forwards the message on the route being deleted. The ROUTE-DELETE message has no effect on the element status in the topology data base since this message only issues on active routes and the element status is not changed from their currently OPERATIVE state.

9. Resource State. Each element (node, link) can have three states: UNKNOWN-ASSUMED OPERATIVE (UNK), OPERATIVE (OP), or INOPERATIVE (INOP). If an element is in the OP or UNK state, it can be utilised for route selection purposes. In the INOP state, the element cannot be utilised for route selection until it is marked UNK or OP.

a) UNKNOWN-ASSUMED OPERATIVE STATE SETTING

1) All elements are considered UNK when the topology data base is activated, as is a new element when added to an already existing topology data base.

2) An element in the INOP state is changed to UNK if another element on the same INOP route but closer to the topology data base also fails. This is done to ensure that an element does not stay INOP indefinitely because a ROUTE-ELEMENT-OPERATIVE report could not reach the topology data base due to a second outage.

b) OPERATIVE STATE SETTING

1) elements change from UNK to OP when a ROUTE-SETUP-REPLY message is received.

2) Elements change from INOP to OP when an element which prevents the success of a ROUTE-SETUP request becomes operative, causing the generation of a ROUTE-ELEMENT-OPERATIVE message.

c) INOPERATIVE STATE SETTING

1) A ROUTE-FAILURE resulting from a ROUTE-SETUP attempt causes a state change to INOP for the identified inoperative element over which the ROUTE-SETUP could not be forwarded.

2) A ROUTE-FAILURE notification on an active route causes a state change to INOP for the identified

FIG.6

REQUEST CODE	ORIGIN NODE	DESTINATION NODE	CLASS OF SERVICE
78	79	80	82

76

FIG.7

REQUEST CODE	ROUTE ID	NODE	LINK	NODE	LINK	- - -	NODE

86   87   88

84

FIG.8

LPID	OUTBOUND LINK	NEXT LPID
2	1	5

92   96   98

88

90
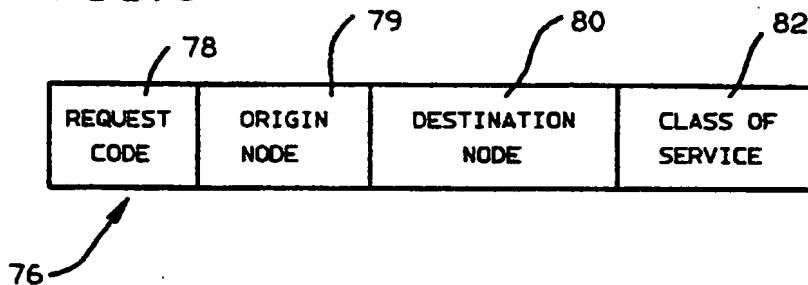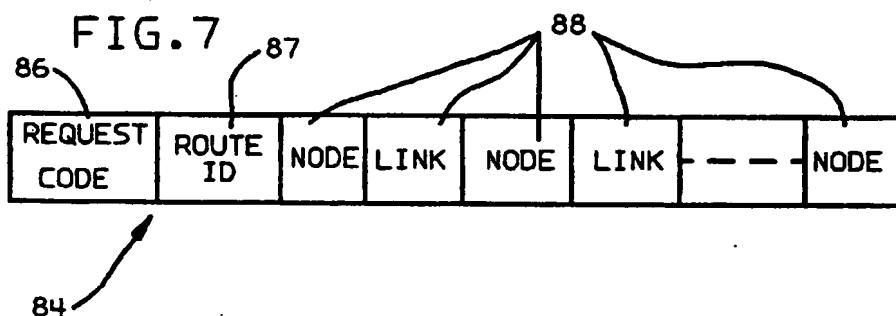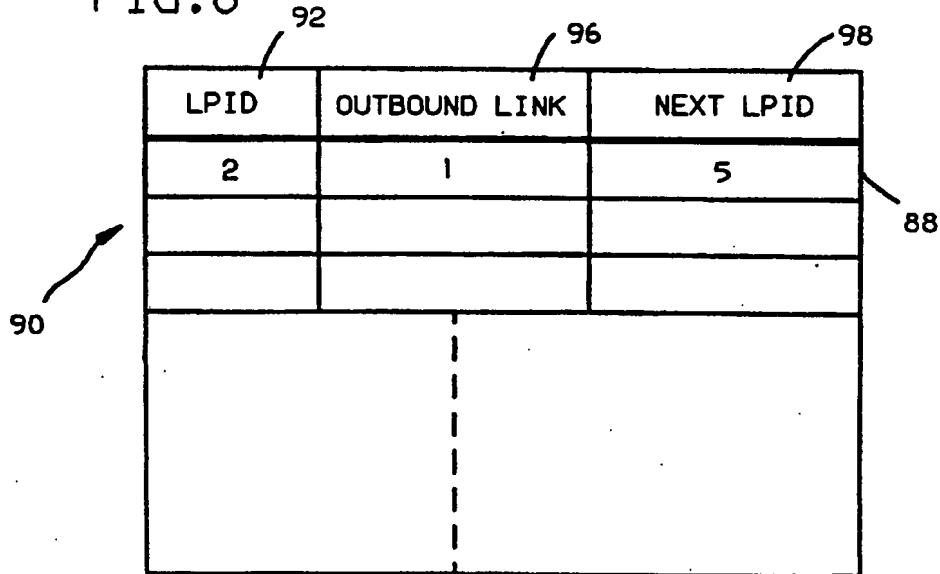
FIG.9
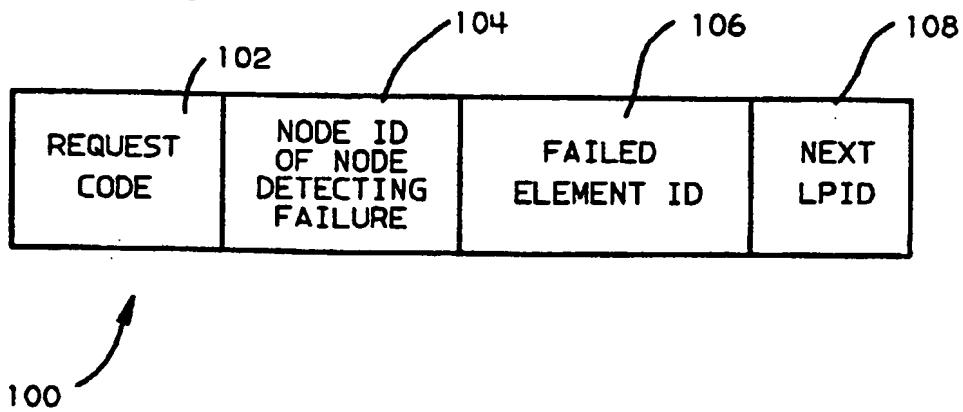
FIG.10

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


       ####   ### ###                              ##
        #      #   #                               #
        #      #   #    ####    ## ###    ### #   ### ###
        #      #####   #   #    ##        #   ##   #   #
  #   #  #   #   #   #  #####    #         #   #    #   #
  #   #  #   #   #   #  #   #    #         #   #    # #
  #   #  #   #   #   #  #  ##    #         #   ##    # #
   ###   ### ###   ### ##  #####     ### ##      #
                                                 #
                                                ###


       ####   ### ###                              ##
        #      #   #                               #
        #      #   #    ####    ## ###    ### #   ### ###
        #      #####   #   #    ##        #   ##   #   #
  #   #  #   #   #   #  #####    #         #   #    #   #
  #   #  #   #   #   #  #   #    #         #   #    # #
  #   #  #   #   #   #  #  ##    #         #   ##    # #
   ###   ### ###   ### ##  #####     ### ##      #
                                                 #
                                                ###


* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```
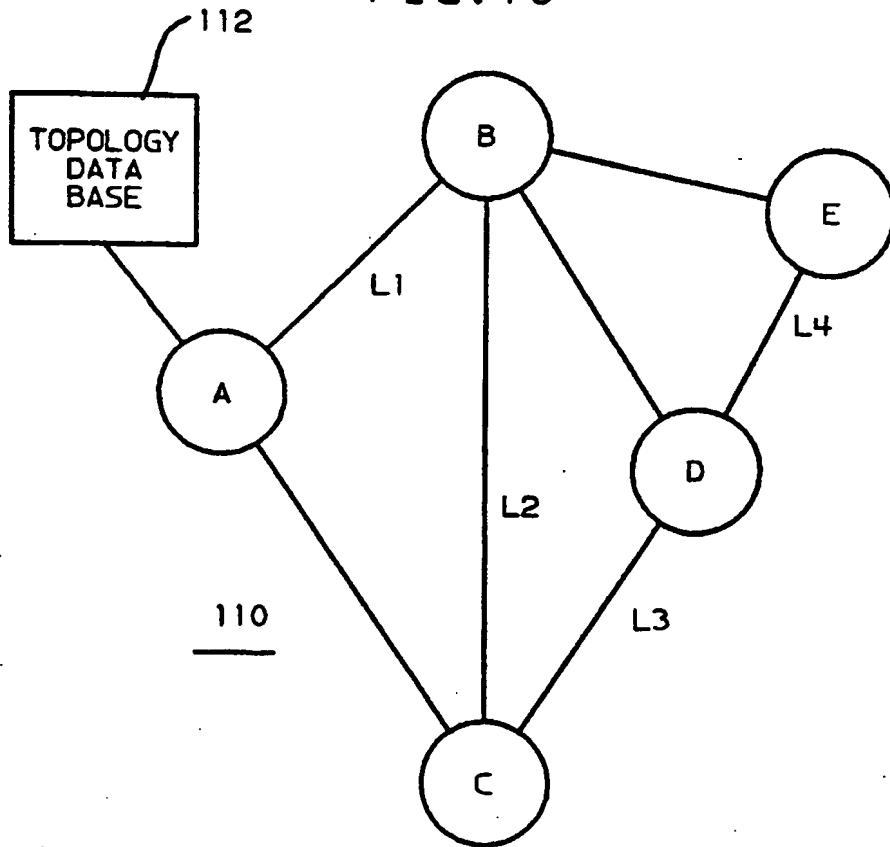
Print Job Information:

Date: 10/22/2007

Time: 1:27:44 PM

Job Number: 144

EUROPEAN PATENT APPLICATION

(71) Applicant: AMT(HOLDINGS) LIMITED
65 Suttons Park Avenue
Reading Berkshire RG6 1AZ(GB)

(72) Inventor: Hunt, David John
3 Moores Green
Wokingham, Berkshire, RG11 1QG(GB)

(74) Representative: Rackham, Stephen Neil et al
GILL JENNINGS & EVERY 53-64 Chancery
Lane
London WC2A 1HN(GB)

(54) Processor array system.

(57) A processor array employs an SIMD architecture and includes a number of sub-arrays (S1...S4). Each sub-array (S1...S4) includes n processor elements (PE). Each processor element is connected to local store including on-chip memory. Each sub-array is connected to a region of off-chip memory by an m-bit wide path, where m is an integer greater than 1. The m-bit wide path is selectively configurable as a one-bit path to or from each of m processor elements or as an m-bit wide path arranged to communicate complete m-bit words of memory data between the region of off-chip memory and respective processor elements.

EP 0 428 327 A1

PROCESSOR ARRAY SYSTEM

The present invention relates to parallel processing computer systems, and in particular to a system comprising san array of processor elements employing an SIMD architecture. One example of such a system is described and claimed in GB-A-1445714, assigned to the present applicants.

5 It is known to construct the array of processor elements from a number of sub-arrays or modules formed on separate chips. Each sub-array comprises a number of processor elements and data paths are provided for the communication of data between neighbouring processor elements both within the sub-array and from one sub-array to adjacent sub-arrays. Each processor element has local store associated with it. Part of this local store is provided as on-chip memory integrated with the processing elements on the sub-array. In addition, in order further to increase the memory available to the processor elements without 10 reducing the levels of integration possible in the chip, off-chip memory is used. The off-chip memory is physically separate from the chip bearing the respective sub-array but is provided with connections to all the processor elements on the sub-array so that each processor element sees a region of the off-chip memory as an extension of its local store.

According to the present invention, in a processor array employing an SIMD architecture, the array 15 comprising a number of sub-arrays and each sub-array comprising n processor elements, each processor element being connected to local store comprising on-chip memory, each chip is connected by an m-bit wide path, where m is an integer greater than 1, to a region of off-chip memory, this path being selectively configurable as a one bit path to or from each of m processor elements, or as an m-bit wide path arranged to communicate complete m-bit words of memory data between the region of off-chip memory and 20 respective processor elements.

Preferably m is equal to n and each sub-array is formed on a separate chip.

In conventional processor arrays built from single bit processors, the region of memory associated with each individual processor is also one bit wide so the successive bits of a given data word or number are held in different locations within that memory region. This is known as a 'vertical' storage mode. At a given 25 time every processor accesses the same bit of the data held in its own memory, this set of bits being referred to as a memory plane. For example, in a particular instruction each processor may access the sign bit of a number. However, when the memory of such a processor array is accessed by the MCU, it normally accesses the memory bits that comprise a particular row of a particular memory plane. Thus a word of data written by the MCU is said to be in 'horizontal' storage mode. If such data is to be processed by the 30 processor array it is usually necessary to re-arrange the data into the 'vertical' storage mode described above, the conversion of data between the two storage modes being referred to as 'corner turning'. Such corner turning can be performed by an instruction sequence involving shifting and merging of data, but there is clearly a performance overhead associated with the input of data to the processor array or the return of results from it.

35 When the processors are byte wide as disclosed in our co-pending application European application, agent's reference 80/3430/02, claiming priority from British application number 8925720.8 and also entitled Processor Array System, the on-chip memory is also byte wide, and successive bytes of each number are similarly held at successive byte wide locations in the memory of a given processor. This may be considered to be a vertical arrangement of bytes, although the bits within each byte are of course accessed 40 and processed in parallel.

Although, as mentioned above, off-chip memory has been used in the past as an extension of the local store the organisation adopted for the off-chip memory has always been the same as that used in the on-chip memory, with the data stored in vertical mode. The present inventor however has discovered that significant advantages can be gained by arranging the off-chip memory in horizontal mode with a word 45 length related to the number of processing elements in the sub-array. Corner turning may then be carried out in transit between the off-chip memory and the processing element using an n-bit shift register.

With the arrangement of the present invention the MCU, that is the scalar processor which controls operation of the array, is able to access an element of a matrix stored in off-chip memory with the same speed as access to a vector element or to a scalar. This provides a significant improvement in the 50 performance of the system when handling matrices.

Preferably the n-bit wide path is also configurable to communicate words of length n/2 or n/4 between the off-chip memory and respective processor elements. Preferably each sub-array is arranged to provide a locally generated address for each word accessed in the off-chip memory.

Preferably the processor array system is connected to a host processor arranged to address the processor array as an extension of its own memory.

When the processor array system is used with a host processor that host processor will generally store data in horizontal mode. Accordingly, providing the sub-array with off-chip memory which similarly uses horizontal mode avoids the need for mode conversion when data is sent from or returned to the host.

5   With known arrays of single bit processors the off-chip memory address is the same for every processor.

In the present invention, each n bit word accessed in off-chip memory may be associated with a single processor, as already explained. The chip is provided with means whereby the address for each such word may optionally be computed locally within that chip rather than being broadcast globally. This technique is known as 'local indexing', since each processor may thereby supply an address for its own data access.

10   This markedly improves the efficiency of certain categories of operation such a table look up and also, for example, helps with shifting of sheet-mapped oversized arrays.

Preferably the local address generation means include an on-chip address buffer arranged to store one address for each processing element.

Preferably each processing element is arranged to construct a local index address in an associated

15   register and to transfer the address from the associated register to the on-chip buffer to write data to a locally-indexed position in the off-chip memory.

Preferably the associated register is an operand register connected to the arithmetic unit of the respective processing element.

Alternatively, each sub-array may include an on-chip buffer for off-chip memory, the on-chip buffer

20   being arranged to hold memory data words associated with respective processor elements and the operand register of each processor element being arranged to hold an associated local memory address.

Preferably the arithmetic unit is a byte-wide processor and the operand register forms part of a multi-byte shift network having a data output for each byte position and the processing element further includes a multiplexer arranged to communicate data from a selected one of the outputs to the arithmetic unit.

25   Where a byte or multi-byte shift register is already present in each processing element functioning as an operand register for the ALU, this same register may be used to receive the data words from the off-chip memory, thereby enabling the required functions to be implemented without any further increase in the complexity of the processing element. The use of a shift-register in conjunction with the ALU, and in particular in conjunction with a byte-wide ALU is described in our above cited co-pending application.

30   A processor array in accordance with the present invention will now be described in further detail with reference to accompanying drawings in which:

Figure 1 is a block diagram of a processor array system;

Figure 2 is a diagram of a processing element for use in the system of figure 1;

Figure 3 shows one configuration of memory interface for a single sub-array.

35   Figure 4 shows an alternative configuration of memory interface;

Figure 5 shows a detail of the memory interface when two 16 bit data words are packed into each memory word.

The system of the present invention is described below in the context of a processor array system in which the processing elements include a byte-wide arithmetic unit and a multi-byte shift register. The

40   system of the present invention is however by no means limited to use with this form of system and may, for example, be used with an otherwise conventional array system such as that described and claimed in the above cited patent.

In the present example each module S1-S4 comprises 32 processing elements PE. Each module has associated with it a block of off-chip memory.

45   The structure of one processing element is shown in Figure 2. The processing element is arranged to operate on a byte of 8 bits. The processing element includes an 8 bit wide arithmetic unit ALU and 8 bit wide data paths for carrying data between the arithmetic unit ALU and on-chip memory. As described in further detail below, the processing element further includes a four byte wide 32 bit operand shift network Q comprising a byte-wise shift network Q1, a bit-wise shift network Q2 and an output register Q0. The shift

50   networks are formed from appropriately interconnected 2:1 multiplexers. The output from each byte position of the output register Q0 is connected to the corresponding input of the byte-wise shift network Q1 to provide a cyclical data path. Another connection from each output of the output register Q0 goes to a multiplexer MUX which is linked by a byte-wide data path to an input of the arithmetic unit ALU to communicate data from a selected one of the four byte positions of the operand shift network Q. The output

55   of the arithmetic unit ALU is input at one end of the byte-wise shift network Q1 and the single bit carry-output of the arithmetic unit ALU is input at one end of the bit-wise shift network Q2. The data output from the other end of the bit-wise shift register Q2 is taken via a bit-wide data path to one end of an 8-bit shift register S, the "multiplier" register.

3

The different elements of the processing element PE and examples of the use of the processing element PE in carrying out arithmetic operations are described in further detail below.

Memory Paths

This is organised as 8 bits wide and a byte might be read or written in one clock cycle but not both in the same cycle. In practice it is convenient to have at least 512 bits of on-chip memory per PE. For clarity, the on-chip memory is omitted from Figure 2. The read port of the on-chip memory is shown at the top of the figure, and the write port at the bottom.

The PE data paths for the least significant bit are substantially the same as in conventional single-bit processing elements, such as those described in the above cited patent. However for a full set of single-bit operations on the on-chip memory the following additional functions are necessary;

1. On the read port, option to select any bit of the on-chip memory byte and place it on the least significant bit. The other bits on the PE input can be regarded as undefined.

2. On the write port, option to replicate the least significant bit of the ALU output in all the bits. Alternatively this replication may be done in the ALU itself.

3. Option to write just one bit of the byte into on-chip memory. Preferably this is done by gating the writes to individual bits; it may alternatively be done as read-merge-write but this takes longer.

These additional functions listed above are equivalent to providing single-bit access (as well as byte access) to the on-chip memory. In general the bit address used for accessing the memory is different from that used for masking the ALU, since for single bit operation it is always the least significant bit of the ALU that is used.

A one bit wide path from off-chip memory is multiplexed with the least significant bit of the on-chip memory, the other bits as being "don't care" in this case. Similarly, off-chip memory may be written from the least significant bit of the on-chip memory write path.

Q Register

As described above, in the present example the Q register is a 32 bit wide register with shift facilities. In general the length of the Q register is preferably, but not necessarily, matched to the word length of the operands. If, for example, the PE is required to process 48-bit words then Q is preferably at least 48 bits long. It operates like a specialised on-chip memory with simultaneous read and write ports.

It is possible to construct the Q register so that only the least significant byte of the register is used as an ALU operand but input can be made at any byte position. In practice however it is found to be more useful to be able to select any byte as ALU operand but to restrict input to one byte of the register. The Q register may considered to be of length 8, 16, 24 or 32 bits. When emulating the instruction set of existing single-bit arrays, the least significant bit of the most significant byte is used.

The current outputs of the Q register pass in succession through two shift networks:

1. An optional right shift of 8 bits (1 byte) with the ALU output fed in at the MS end. It may be useful also to be able to input the ALU data at the MS byte and pass the other bytes unchanged.

2. An optional right shift of one bit with either the ALU carry output or the value of the register fed in at the MS end. When this is done, the bit shifted out on the right is available to shift in to the S register, i.e. the multiplier register.

The output of the second shifter is always clocked into the Q register. The two shifts may be applied separately or together and either or both may be global or under local activity control.

S Register (Multiplier Register)

The S register is an 8-bit shift register used to hold "old" memory contents for read/write. It is also available as a programmer-visible register used in particular in multiplication. It may be shifted one bit to the right, usually as an extension to the Q register. The least significant bit may be used as a multiplier bit.

Neighbour Input Multiplexer

In general, each processing element will have a data input from each of its four nearest neighbours. The input in general will be one bit wide. The neighbour input multiplexer is not part of the general input multiplexer for each PE as in previous single bit PE designs, but is moved to the carry-in of the arithmetic unit, where the neighbour values are multiplexed with the single bit C register. This arrangement gives greater flexibility of carry propagation than in single bit PEs.

Input Multiplexer

The general input multiplexer functions in a fashion similar to that used in earlier 1-bit arrays, such as that described and claimed in co-pending British application no. 8829622.3 assigned to the present applicant. The multiplexer provides the following options;

1. Q Register. The least significant bit of a selected byte of the Q register on the least significant bit, and zero, i.e. no input, on the other bits. The byte selected for this purpose is the same as is selected for the other ALU input. For response instructions, the most significant byte of Q will normally be used.

2. A register. The inverse of the one-bit register on the least significant bit and zero (i.e. no input) on the other bits. The inverse rather than the non-inverted value is used.

3. Memory input. This is 8-bit data from the on-chip memory read port including the options of either off-chip memory data or a selected bit of the on-chip memory byte, on the least significant bit.

4. S Register. 8 Bits.

5. Row/Column data from the MCU. The least significant bit receives single bit data broadcast from the MCU in the row or column direction. The arrangement is described in our earlier application No. 8829622.3 (AMT Case 2003). This least significant bit may be data written into the PE or memory, or a mask used in row or column selection in the response data. For the latter purpose it is ORed with some other input to the multiplexer. The other bits receive 7-bit data common to all other PE's broadcast from the MCU. Thus an eight bit literal value can in general be used as operand.

6. D Register. The one-bit D register on the least significant bit and zero (i.e. no input) on the other bits. The D register is arranged in a conventional fashion to permit data to be shifted into or out of the array at the same time as processing functions are being performed and is not shown in Figure 2.

The row/column response is taken from the least significant bit of the input multiplexer.

A Register

This functions in a manner similar to that described in the above cited application. The register is loaded from the carryout of the ALU either directly or ANDed ("masked") with the existing A register value. Compatibility with the functions of a conventional single bit array may be provided by generating the appropriate functions for the A register and the carry out of the ls bit of the ALU and masking off the other ALU bits, so that the required value propagates through to the carry output. This arrangement permits also setting activity according to the results of arithmetic tests.

Activity Select

Activity control can apply to the on-chip memory, the off-chip memory, or the shifting or loading of the Q register. At the level of each individual PE, activity control is identical to the corresponding function described in the above cited patents and applications. It provides options for activity to be equal to A or its inverse or to be controlled on a row or column basis. A further level of masking control may be applied to the individual bits of on chip memory. The set of 8 mask bits is common to all PE's and is AND-ed with the above control. This gives the option of writing to an individual bit of a byte or in general to specify a particular bit field within the byte. The mask pattern is generally the same pattern that is used to control the ALU function unless single bit writes to on-chip memory are done by masking. In the present example, activity can also be set according to the ls bit of the S register. As described below, this helps the implementation of the "multiply" function.

C Register

This is one bit wide and may be loaded from the carry borrow output of the ALU. It may be used as carry in to the ALU or as a serial shift input for the Q register.

5 ALU

As described above, this is now 8 bits wide, taking a selected byte of the operand register means Q and of the input multiplexer data as input and the C register as carry-in.

A variety of functions are provided. To give maximum flexibility of single bit functions the least 10 significant bit has full function units for both sum and carry outputs. For the other bit positions arithmetic add, subtract and reverse subtract are provided, together with copy of either operand, and a variety of bit-by-bit boolean functions.

The ALU has masking features which permit operations on selected bit fields rather than being constrained to byte boundaries. Thus an 8-bit mask, common to all PE's is applied to the ALU and has the 15 following effect:

If mask bit = 1 normal generation of carry out bit

If mask bit = 0 carry out bit = carry in bit for that bit position.

Result bit: If mask bit = 1 the normal result bit function for that bit is provided at the ALU output. If mask bit = 0 the result value is of no interest, so any value convenient to the implementation may be 20 provided.

Masking the ALU operation may be implemented by explicitly gating the carry propagation in the manner described above, but this may preclude the use of standard carry predict techniques for fast operation of the ALU. An alternative is to achieve the masking by forcing one input of the ALU to 1 and the other to 0, for each bit position where the mask bit is 0.

25 The mask pattern comprises a consecutive set of truebits specified by a start bit and an end bit. This of course allows selection of a single bit. For operations on a complete byte, the start bit is specified as zero and the end bit as 7. For single-bit operations the least significant bit of the ALU is used, so the start and end bits are both specified as 7. Thus, the carry out of the least significant bit is propagated unchanged through the other ALU bits and may be clocked into C.

30 During multiply the LS bit of the S register acts as a multiplier and may achieve this by selecting the ALU function as either "copy" or "add" dependent on the local value of that bit. There is also provision for local control to distinguish between add and subtract, for such purposes as non-restoring division. The LS bit of the S register may be used to control this function.

35

Merge Function

This is the logic that selects the memory output data as either the ALU output or the old memory contents. It is 8 bits wide and each bit has its own activity select but its function is otherwise the same as in 40 the single bit systems disclosed in the above cited patents and applications. The merge function gives an 8 bit path to the on-chip memory and as already noted the off-chip memory path is taken from the least significant bit.

45 Neighbour Paths

The neighbour output (not shown in the Figure) needs to be either the least significant bit of a selected byte of the Q register (for shift functions) or the carry- out of the ALU (for ripple add functions). The memory path may be common with the neighbour output. The selected neighbour input may be used 50 instead of the C register as the carry-in to the ALU.

D Plane

55 As in the systems described in the above cited patents, a D-register forming a data plane for Fast Input-Output operations may be used. The D register can be loaded from memory input, or supplied as data on the memory write path. A single bit D register may be used, linked only to the off-chip memory. If the on-chip memory is operated as a cache then I/O data can be cached and D plane transfers to or from

Fig.4.
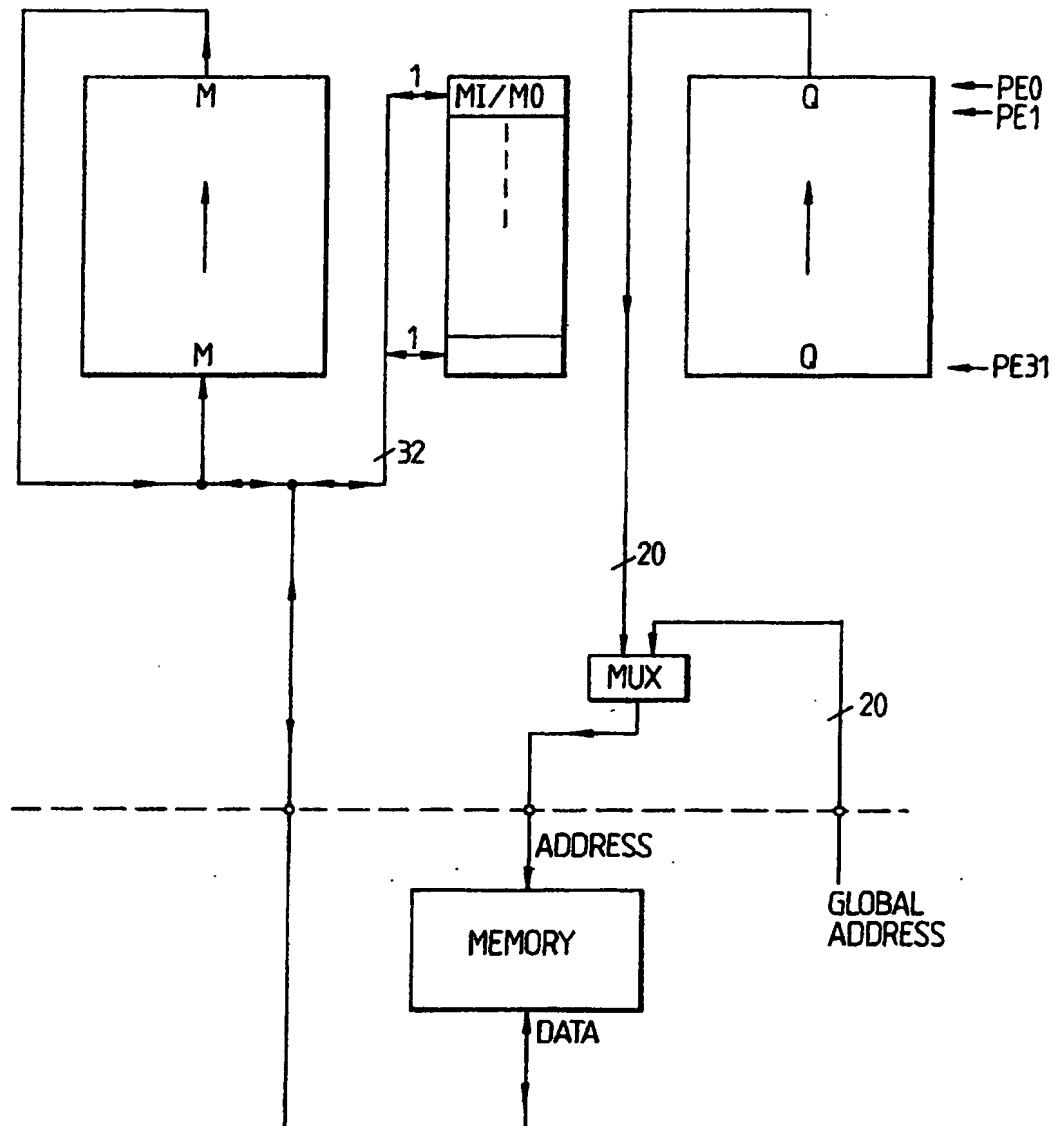
Fig.5.
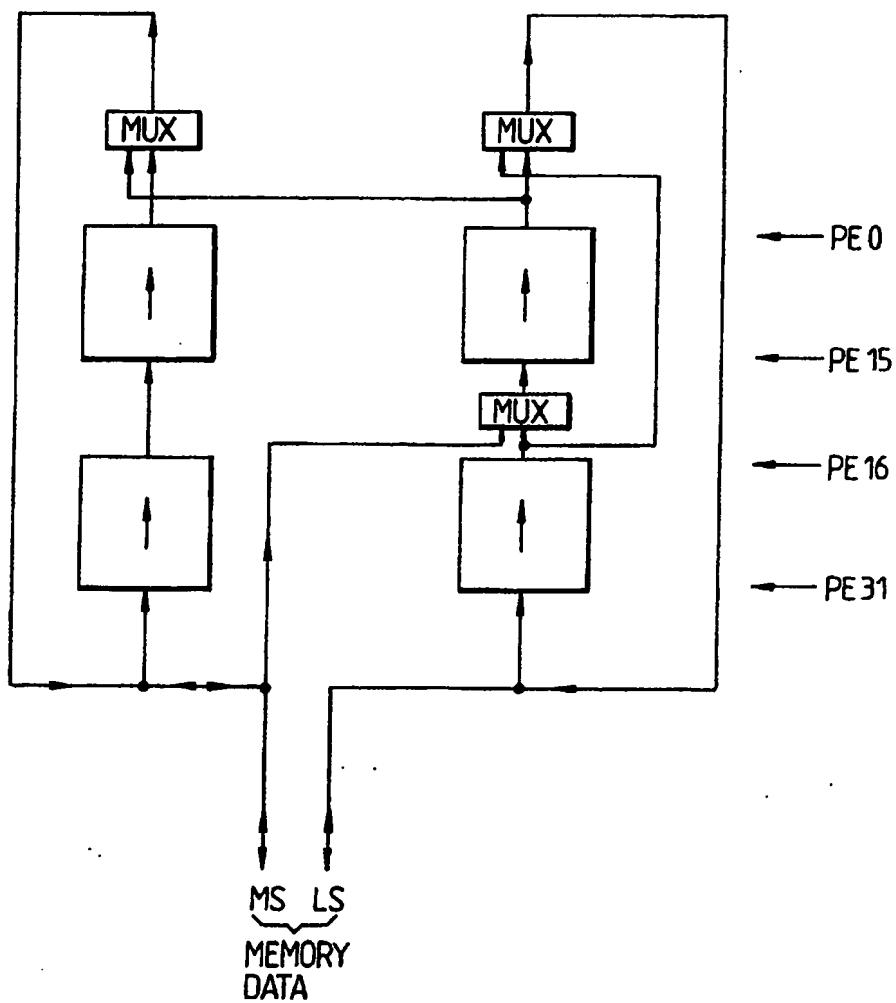
MS LS

MEMORY
DATA

PE 0

PE 15

PE 16

PE 31

EUROPEAN SEARCH REPORT

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (int. Cl.5)
A	EP-A-0 191 280 (IBM) * Abstract * — — —	1,2,3	G 06 F 15/80
A	US-A-4 144 566 (THOMSON-CSF) * Figures 5,6; abstract; column 5, lines 65-69; column 6, lines 1-69 * — — — — —	8,9	

TECHNICAL FIELDS SEARCHED (int. Cl.5)
G 06 F 15 G 06 F 12

The present search report has been drawn up for all claims

Place of search	Date of completion of search	Examiner
The Hague	10 December 90	BLIONAS S.